REMARKS

The Office Action mailed June 29, 2006 has been reviewed and carefully considered. New claims 11-16 have been added. No new matter has been added. Claims 1-16 remain pending in this application with claims 1, 5 and 9 being the only independent claims.

Reconsideration of the above-identified application in view of the following remarks is respectfully requested.

For the foregoing reasons applicant submits that the claims are patentable over the prior art of record and requests that the application be passed to issuance.

Claims 1-10 are rejected under 35 U.S.C. §102(b) as anticipated by Bahrs et al. (U.S. Patent No. 6,002,874). In addition, claims 1-10 are rejected under 35 U.S.C. §102(e) as anticipated by Tondreau et al. (U.S. Patent Application Publication No. 2003/0226123). Claims 1-10 are rejected under 35 U.S.C. §102(e) as anticipated by Dupuy et al. (U.S. Patent No. 6,523,171).

Applicants respectfully traverse the prior art rejections for the reasons described in detail below.

I. Claim 1-10 Anticipated by Bahrs et al.

Independent Claim 1

Claim 1 calls for a method for universal programming language conversion between two different sequential programming languages including a source program in a first programming language and a target program in a second programming language. To eliminate the possibility of syntax related conversion errors the present inventive method includes the steps of "stripping all syntax from the parsed source program"; "receiving as input the parsed source program without any syntax"; "instantiating classes in a framework for capturing semantics of the parsed source program independent of syntax and execution model of the sequential programming languages"; and "producing a semantic representation of the parsed source program without any syntax" . Thus, the claim expressly calls for all syntax to be preliminarily stripped so that thereafter the instantiation of classes in a framework for capturing semantics is independent of syntax.

In rejecting the claimed invention the Examiner maintains that Bahrs et al. reads on these limitations (Col. 2, ll. 28-51). "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). Bahrs et al. discloses a method and system for translating goto-oriented procedural languages into goto-free object oriented languages. Specifically, Bahrs et al. recognizes that object oriented programming languages do not employ "goto" branch constructions. Accordingly, Bahrs et al. eliminates the "goto" statements when transforming an existing procedural source code that includes such statements to an object oriented language that has structured programming control flow constructs rather than "goto" branch constructions. Thus, in Bahrs et al., the only focus or concern when transforming from a procedural source program to a target object oriented program is the "goto" statement.

The Examiner asserts that the conversion step of Bahrs et al. (Col. 2, ll. 28-45) inherently provides for stripping to get rid of the goto structure. Applicants respectfully traverse the Examiner's rejection and assert that the overall goal or result of the Bahrs et al. transformation is to eliminate only_the "goto statement". Bahrs et al. fails to disclose or suggest the stripping of <u>all</u> syntax from the parsed source program as expressly called for in claim 1,. Therefore, Bahrs et al. does not anticipate the invention found in claim 1. Not only does Bahrs et al. fail to anticipate claim 1, Applicants submit that no motivation exists for possible modification by the Examiner of Bahrs et al. in combination with another reference to render the present claimed invention obvious under 35 U.S.C. §103(a). "A prior art reference must be considered in its entirety, i.e., as a <u>whole</u>, including portions that would lead away from the claimed invention." *W.L. Gore & Associates, Inc. v. Garlock, Inc.*, 721 F.2d 1540, 220 USPQ 303 (Fed. Cir. 1983), *cert. denied*, 469 U.S. 851 (1984). Bahrs et al. leads away from being modified to strip <u>all</u> syntax from the parsed source program by its intended goal or purpose of elimination of only the goto structure during translation of procedural goto based source code into object oriented target code that is free of goto structure. Such a lack of motivation or suggestion for modification of Bahrs et al. to include the step of stripping of all syntax from the parsed source program would render any potential obviousness rejection by the Examiner improper.

Addressing the instantiating classes step the Examiner asserts that this limitation is disclosed by Col. 2, ll. 38-45 of Bahrs et al. This text from Bahrs et al. reads "generating an output object oriented source program by: generating object class definitions and object methods based on the selected granularity and containing processing logic from the logic analysis; generating object initialization code by creating instances of each object required for program execution; and generating object invocation code for processing the generated objects in accordance with the control flow." However, Bahrs et al. fails to disclose or suggest that the instantiated classes in the framework capture semantics of the parsed source program independent of syntax and execution model of the sequential programming languages, as recited in claim 1.

The next limitation of claim 1 provides "producing a semantic representation of the parsed source program without any syntax". The Examiner refers to the levels of granularity disclosed by Bahrs et al. as teaching this limitation. Bahrs et al. discloses that objects resulting from the code transformation can be at varying levels of granularity. Specifically, "at the finest level of granularity each statement in the procedural language code can be transformed into a Java object. At the other end of the continuum, each procedural program as a whole can be transformed into a single object. In between these extremes, the procedural source code may be transformed into object oriented code with each paragraph comprising an object or, in certain circumstances, each file of source code comprising an object." (Col. 4, ll. 10-18) Therefore the granularity is simply the grouping of procedural language code to be transformed. Irrespective of the particular level of granularity the only syntax to be eliminated is "goto statements". Accordingly, Bahrs et al. fails to disclose or suggest producing a semantic representation of the parse source program without any syntax, as found in claim 1.

The sixth step of claim 1 calls for "receiving the semantic representation at a printer interface specific to the second programming language". The Examiner states that this limitation is taught by Col. 6, ll. 18-24 of Bahrs et al. that reads "The transformation means 104 analyzes the source code 102 and locates each instance of a particular statement type. As the transformation means generates output code, it will create an instance of each particular type supplying the appropriate parameters as extracted from the original procedural code." However, this passage is directed to the transformation means 104 (Figure 1) and nothing in this passage or the remainder of Bahrs et al. discloses that the transformation means 104 includes a "printer interface", as further defined claim 1.

It is the printer interface that performs the last step of claim 1 of "adding the syntax of the target program in the second programming language". The Examiner maintains that this limitation is taught by Col. 5, ll. 9-26 of Bahrs et al. However, the Bahrs et al. reference fails to disclose or suggest adding back into the target program in the second programming language syntax that had previously been stripped out. Accordingly, Applicants assert that Bahrs et al. fails to anticipate claim 1. Furthermore, it would not be obvious to modify the reference to include a printer interface for adding the syntax of the target program in the second programming language since Bahrs et al. expressly teaches away from such modification. "If a proposed modification would render the prior art invention being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification." MPEP 2143.01 (quoting *In re Gordon*, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984)). To modify the Bahrs et al. reference to read on the present claimed invention would be contrary to the expressed goals and teachings of Bahrs et al., which is eliminating all "goto statements" in the object oriented target program. Adding back into the target program the "goto statements" that were previously eliminated would defeat the entire purpose of the Bahrs et al. patent. Accordingly, Bahrs et al. fails to not only anticipate but also render obvious in combination with any other reference the present claimed invention.

Dependent Claim 2

Claim 2 depends from independent claim 1 and thus is patentable over the prior art of record for at least the reasons expressed above with respect to independent claim 1.

Dependent Claim 3

Claim 3 states that "the source program is a high level programming language and the target program is a low level programming language". The terms "high level programming language" and "low level programming language" are well know in the art. A low level programming language is a machine or assembly language. In rejecting these claims the Examiner asserts that Col. 1, ll. 44-48 reads on this limitation. The relevant passage cited by the Examiner reads "The following description will focus on transforming existing programs to programs written in the Java language but there is no restriction on the application of this invention to generation of other object oriented languages such as C++ and Small Talk." The Bahrs et al. invention is expressly disclosed for transforming a "goto" oriented procedural language into a goto-free object oriented language as the target program. An object oriented language is high level programming language, rather than a low level programming language. Therefore, Bahrs et al. fails to disclose or suggest conversion to a target program in a low level programming language.

Not only does Bahrs et al. fail to anticipate claim 1, but the reference expressly teaches away from any potential modification of Bahrs et al. in combination with a secondary reference to render the claimed invention obvious under 35 U.S.C. §103(a). "If a proposed modification would render the prior art invention being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification." MPEP 2143.01 (quoting *In re Gordon*, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984)). The expressly disclosed intended purpose of the Bahrs et al. reference is to eliminate the goto structure when transforming a goto based source code into an object oriented source code that does not recognize goto statements. As acknowledged in the Background and Related Art section of Bahrs et al.(Col. 1, ll. 59-67) assembler language utilizes goto statements, hence the modification of Bahrs et al. to transform a procedural source code into an assembly language would defeat the expressly disclosed intended purpose of elimination of goto structure in the target program that does not recognize such a construction.

Dependent Claim 4

Claim 4 requires the classes to be "C++ classes representing fundamental core constructs of all sequential programming languages". The Examiner asserts that this limitation reads on Col. 1, ll. 44-49 of Bahrs et al., which reads "The following description will focus on transforming existing programs to programs written in the Java language but there is no restriction on the application of this invention to generation of other object oriented languages such as C++ and Small Talk." This passage merely discloses that the Bahrs et al. patented conversion is applicable for a target program in any object oriented programming language, an example of which is C++. However, Bahrs et al. fails to disclose or suggest that such classes are defined to represent "fundamental core constructs of all sequential programming languages", as found in claim 4.

Apparatus Claims 5-8

Claims 5-8 are the apparatus counterparts of claims 1-4 and thus distinguishable over the prior art of record for at least the same reasons discussed above with respect to claims 1-4.

Independent Claim 9

Apparatus claim 9 calls for "a processor for instantiating classes in a framework representing a unification of semantics of the sequential programming languages independent of syntax and execution model". This apparatus limitation is similar to the counterpart instantiating step in claim 1 and thus distinguishable over Bahrs et al. for at least the same reasons expressed above with respect to claim 1.

Dependent Claim 10

Claim 10 depends from independent claim 9 and thus is distinguishable over the prior art of record for at least the reasons expressed in the preceding paragraph with respect to independent claim 9.

New Dependent Claims 11-16

Claims 11-16 call for the first and second programming languages to be either both procedural programming languages or both object oriented programming languages. In contrast, as acknowledged by the Examiner Bahrs et al. discloses transformation from a source program in a procedural programming language to a target program in an object oriented programming language. Applicants therefore assert that Bahrs et al. fails to anticipate claims 11-16.

In the alternative, Applicants submit that no motivation exists for modifying Bahrs et al. in combination with another reference so that the first and second programming languages are either both procedural or both object oriented to possibly render claims 11-16 as obvious under 35 U.S.C. §103(a). "If a proposed modification would render the prior art invention being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification." MPEP 2143.01 (quoting *In re Gordon*, 733 F.2d 900, 221

USPQ 1125 (Fed. Cir. 1984)). The expressly disclosed intended purpose of the Bahrs et al. reference is to eliminate the goto structure when transforming a goto based source code into an object oriented source code that does not recognize goto statements. If the source and target programs are both either procedural or both object oriented, then, in the former, both languages utilize the "goto structure" whereas in the latter neither language recognizes this construction. Accordingly, when the programming languages are both the same type, there is no need for elimination of the "goto structure" thereby defeating the purpose of Bahrs et al.

II. Claim 1-10 Anticipated by Tondreau et al.

Independent Claim 1

Claim 1 calls for "stripping all syntax from the parsed source program". In rejecting claim 1, the Examiner cites paragraph 0019 of Tondreau et al. as support for the statement that the original code is stored in an intermediate format which "inherently" requires stripping. However, the relevant text from paragraph 0019 only states "storing the scanned procedural language code as intermediate code". Nothing in the cited paragraph or the remainder of Tondreau et al. discloses, suggests or necessarily requires that all syntax be stripped from the parsed source program as part of this storing step. MPEP Section 2112 quoting *Ex parte Levy* provides that "In relying upon the theory of inherency, the examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that the allegedly inherent characteristic necessarily flows from the teachings of the applied prior art." 17 USPQ2d 1461, 1464 (Bd. Pat. App. & Inter. 1990) (emphasis in original). In the rejection, there is no basis in fact or technical reasoning that the step of stripping all syntax from the parsed source program necessarily flows from the disclosed step of storing the scanned procedural language code as intermediate code.

Claim 1 further requires the step of "instantiating classes in a framework for capturing semantics of the parsed source program independent of syntax and execution model of the sequential programming languages". Paragraph 35 of Tondreau et al. has been cited by the Examiner as teaching this limitation. This disclosure contemplates the storing of legacy components as meta-components in a repository. Specifically, the disclosure reads that to transform a legacy language software application into a particular object language requires "a transformation process 156 and a regeneration process 166 capable of <u>creating object classes in the desired object language</u> from the object meta-components" (emphasis added). Similar description is also found in paragraph 34 which states "the object meta-components are implemented as actual object classes in a <u>specific</u> object-oriented language". In contrast to the present claimed method that includes the step of "instantiating classes in a framework for capturing semantics of the parsed source program <u>independent of syntax and execution model of the sequential programming languages</u>", Tondreau et al. discloses the creation of object classes in a <u>specific</u> object-oriented language.

The last two steps of claim 1 are "receiving the semantic representation at a printer interface specific to the second programming language" and "adding the syntax of the target program in the second programming language." Paragraph 0036 referred to by the Examiner describes the capability of adding "customization" steps in the transformation system. However, Tondreau et al. fails to disclose or suggest a printer interface or any other means for adding the syntax of the target program in the second programming language, as found in claim 1. The mere broad statement that additional customization is possible is not a sufficient teaching to anticipate the present claimed invention that expressly calls for "receiving the semantic representation at a printer interface specific to the second programming language" and "adding the syntax of the target program in the second programming language."

<u>Dependent Claim 2</u>

Claim 2 depends from independent claim 1 and thus is distinguishable over the prior art of record for at least the reasons expressed above with respect to independent claim 1.

<u>Dependent Claim 3</u>

Claim 3 states that "the source program is a high level programming language and the target program is a low level programming language". The terms "high level programming language" and "low level programming language" are well know in the art. A low level programming language is a machine or assembly language. In rejecting these claims the Examiner asserts that paragraph 33 teaches this limitation. However, the relevant passage cited by the Examiner describes that the system and method performs a transformation for transforming legacy applications into modern, object-oriented N-tier applications that use a graphical user interface. Thus, Tondreau et al. invention is only suitable for transforming a procedural language into an object oriented language as the target program. An object oriented language is high level programming language, rather than a low level programming language. Therefore, Tondreau et al. fails to disclose or suggest conversion to a target program in a low level programming language. Accordingly, Applicants assert that the Tondreau et al. fails to anticipate claim 3. In addition, no motivation exists for the Examiner to modify Tondreau et al. as taught by another reference so that the target program is a low level program to render claim 3 obvious under 35 U.S.C. §103(a). "If a proposed modification would render the prior art invention being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification." MPEP 2143.01 (quoting *In re Gordon*, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984)). The expressly disclosed intended purpose of Tondreau et al. is a system and method for transforming a procedural program into an object-oriented program (see Abstract). Therefore, no motivation exists for modifying Tondreau et al. to be used for transforming from a high level programming language to a low level programming language, which by definition does not include an object-oriented language.

Dependent Claim 4

Claim 4 requires the classes to be "C++ classes representing fundamental core constructs of all sequential programming languages". The Examiner asserts that this limitation reads on paragraph 33 of Tondreau. However, this passage merely discloses that the Tondreau et al. conversion is applicable for transforming a legacy application into a modern object-oriented application. Tondreau et al. fails to disclose or suggest that the classes in a framework for capturing semantics of the parsed source program independent of syntax and execution mode of the sequential programming languages are C++ classes, much less, that such classes are defined to represent "fundamental core constructs of all sequential programming languages", as found in claim 4.

Apparatus Claims 5-8

Claims 5-8 are the apparatus counterparts of claims 1-4 and thus distinguishable over the prior art of record for at least the same reasons discussed above with respect to claims 1-4.

Independent Claim 9

Apparatus claim 9 calls for "a processor for instantiating classes in a framework representing a unification of semantics of the sequential programming languages independent of syntax and execution model". This apparatus limitation is similar to the counterpart instantiating step in claim 1 and thus distinguishable over Tondreau et al. for at least the same reasons expressed above with respect to claim 1.

Dependent Claim 10

Claim 10 depends from independent claim 9 and thus is distinguishable over the prior art of record for at least the reasons expressed in the preceding paragraph with respect to independent claim 9.

New Dependent Claims 11-16

Claims 11-16 call for the first and second programming languages to be either both procedural programming languages or both object oriented programming languages. In contrast, Tondreau et al. discloses transformation from a source program in a procedural programming language to a target program in an object oriented programming language. Applicants therefore assert that claims 11-16 are not anticipated by Tondreau et al.

Moreover, no motivation exists for the Examiner to modify Tondreau et al. as taught by another reference so that the target program is a low level program to possibly render claims 11-16 obvious under 35 U.S.C. §103(a). "If a proposed modification would render the prior art invention being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification." MPEP 2143.01 (quoting *In re Gordon*, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984)). The expressly disclosed intended purpose of Tondreau et al. is for transforming a procedural program into an object-oriented program (see Abstract). Therefore, no motivation exists for modifying Tondreau et al. to be used for transforming from a procedural programming language to a procedural programming language or from an object-oriented programming language to an object-oriented programming language.

Accordingly, there is no motivation and thus no basis for claims 11-16 to be rejected under 35 U.S.C. §103(a) as obvious over Tondreau et al.

III. Claims 1-10 Anticipated by Dupuy et al.

Independent Claim 1

Claim 1 comprises the step of "stripping all syntax from the parsed source program". Dupuy et al. (Col. 2, ll. 4-29) was referred to by the Examiner as teaching this limitation. However, this passage describes grouping of individual characters into tokens which is the explanation of lexical analysis rather than stripping of all syntax from the parsed source program, as in claim 1.

In addition, method claim 1 further includes the step of "instantiating classes in a framework for capturing semantics of the parsed source program independent of syntax and execution model of the sequential programming language". To the contrary, the Java classes created in Dupuy et al. are not instantiated. Furthermore, the Java classes in Dupuy et al. are generated dependent on the target programming language (Col. 4, ll. 23-25), whereas in the present claimed invention the classes in a framework for capturing semantics of the parsed source program are independent of syntax and execution model of the sequential programming language.

The last two steps in claim 1 state "receiving the semantic representation at a printer interface specific to the second programming language" and "adding the syntax of the target program in the second programming language". In rejecting claim 1, the Examiner states that these limitations are taught by Dupuy et al. (Col. 4, ll. 11-25 and steps (c) & (d) of claim 1). However, these passages of Dupuy et al. disclose the creation of target object oriented classes rather than a printer interface or any other means for adding the syntax of the target program in the second programming language, as in claim 1.

Dependent Claim 2

Claim 2 depends from independent claim 1 and thus is distinguishable over the prior art of record for at least the reasons expressed above with respect to independent claim 1.

Dependent Claim 3

Claim 3 states that "the source program is a high level programming language and the target program is a low level programming language". The terms "high level programming language" and "low level programming language" are well know in the art. A low level programming language is a machine language or assembly language. A high level programming language can include a procedural language or an object oriented language, but does not include a machine language or assembly language. In rejecting this claim the Examiner asserts that the abstract teaches the above limitation. However, the abstract describes translating a program written in a procedural programming language to a program written in an <u>object oriented language, both of which are high level programming languages</u>. Therefore, Dupuy et al. fails to disclose or suggest conversion to a target program in a low level programming language. Accordingly, Applicants assert that the Dupuy et al. fails to anticipate claim 3.
In addition, no motivation exists for the Examiner to modify Dupuy et al. as taught by another reference so that the target program is a low level program to render claim 3 obvious under 35 U.S.C. §103(a). "If a proposed modification would render the prior art invention being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification." MPEP 2143.01 (quoting *In re Gordon*, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984)). The expressly disclosed intended purposes of Dupuy et al. is translating source code programs written in a procedural computer language to source code programs written in an object oriented language (see Abstract). Therefore, no motivation exists for modifying Dupuy et al. to be used for transforming from a high level programming language to a low level programming language, which by definition does not include an object-oriented language.

Dependent Claim 4

Claim 4 requires the classes to be "C++ classes representing fundamental core constructs of all sequential programming languages". The Examiner once again refers to the Abstract to read on this limitation. However, Dupuy et al. expressly teaches (Col. 4, ll. 7-34) the use of Java rather than C++ classes. In addition, Dupuy et al. fails to disclose or suggest that the classes are defined to represent fundamental core constructs of all sequential programming languages, as in claim 4.

Apparatus Claims 5-8

Claims 5-8 are the apparatus counterparts of claims 1-4 and thus distinguishable over the prior art of record for at least the same reasons discussed above with respect to claims 1-4.

Independent Claim 9

Apparatus claim 9 calls for "a processor for instantiating classes in a framework representing a unification of semantics of the sequential programming languages independent of syntax and execution model". This apparatus limitation is similar to the counterpart instantiating step in claim 1 and thus distinguishable over Dupuy et al. for at least the same reasons expressed above with respect to claim 1.

Dependent Claim 10

Claim 10 depends from independent claim 9 and thus is distinguishable over the prior art of record for at least the reasons expressed in the preceding paragraph with respect to independent claim 9.

New Dependent Claims 11-16

Claims 11-16 call for the first and second programming languages to be either both procedural programming languages or both object oriented programming languages. In contrast, as acknowledged by the Examiner Dupuy et al. discloses transformation from a source program in a procedural programming language to a target program in an object oriented programming language. Applicants therefore assert that claims 11-16 are not anticipated by Dupuy et al.

Moreover, no motivation exists for the Examiner to modify Dupuy et al. as taught by another reference so that the target program is a low level program to possibly render claims 11-16 obvious under 35 U.S.C. §103(a). "If a proposed modification would render the prior art invention being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification." MPEP 2143.01 (quoting *In re Gordon*, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984)). The expressly disclosed intended purpose of Dupuy et al. is for translating a procedural computer program into an object-oriented program (see

Abstract). Therefore, no motivation exists for modifying Dupuy et al. to be used for translating from a procedural programming language to a procedural programming language or from an object-oriented programming language to an object-oriented programming language. Accordingly, there is no basis for rejection of claims 11-16 under 35 U.S.C. §103(a) as obvious over Dupuy et al.

For the foregoing reasons, Applicants submit that claims 1-16 are patentable over the art of record. Withdrawal of the prior art rejections and passage of the application to issuance is therefore requested.

Respectfully submitted,
SCHWEITZER CORNMAN GROSS & BONDELL LLP

By [illegible]
Fritz L. Schweitzer III
Reg. No. 39,363
292 Madison Avenue – 19th Floor
New York, NY 10017
Phone: (646) 424-0770
Fax: (646) 424-0880

FS3/CFC